Complexity of Approximate Query Answering under Inconsistency in Datalog$^{\pm}$

Thomas Lukasiewicz[1], **Enrico Malizia**[2], **Cristian Molinaro**[3]
[1] Department of Computer Science, University of Oxford, UK
[2] Department of Computer Science, University of Exeter, UK
[3] DIMES, University of Calabria, Italy
thomas.lukasiewicz@cs.ox.ac.uk, e.malizia@exeter.ac.uk, cmolinaro@dimes.unical.it

Abstract

Several semantics have been proposed to query inconsistent ontological knowledge bases, including the intersection of repairs and the intersection of closed repairs as two approximate inconsistency-tolerant semantics. In this paper, we analyze the complexity of conjunctive query answering under these two semantics for a wide range of Datalog$^{\pm}$ languages. We consider both the standard setting, where errors may only be in the database, and the generalized setting, where also the rules of a Datalog$^{\pm}$ knowledge base may be erroneous.

1 Introduction

Description logics (DLs) and existential rules from the context of Datalog$^{\pm}$ are popular ontology languages. In real-world ontology-based applications involving large amounts of data (such as ontology-based data extraction and/or integration), it is very likely that the data are inconsistent with the ontology, and thus inconsistency-tolerant semantics for ontology-based query answering are urgently needed.

Consistent query answering, first developed for relational databases [Arenas *et al.*, 1999] and then generalized as the AR semantics for several DLs [Lembo *et al.*, 2010], is the most widely accepted semantics for querying inconsistent ontologies. Query answering under the AR semantics is known to be a hard problem, even for very simple languages [Lembo *et al.*, 2010]. For this reason, several other semantics have been recently developed with the aim of approximating consistent query answering [Lembo *et al.*, 2010; Bienvenu, 2012; Lukasiewicz *et al.*, 2012a; Bienvenu and Rosati, 2013].

In particular, in [Lembo *et al.*, 2010], besides the AR semantics, three other inconsistency-tolerant query answering semantics are proposed, including the approximate *intersection of repairs (IAR)* semantics, in which an answer is considered to be valid, if it can be inferred from the intersection of the repairs (and the ontology). The *intersection of closed repairs (ICR)* [Bienvenu, 2012] is another approximate semantics, in which an answer is valid, if it can be inferred from the intersection of the closure of the repairs (and the ontology).

There are several reasons for the practical relevance of the IAR and the ICR semantics, and thus for motivating an in-depth analysis of their computational properties. First, they are two natural semantics that identify "surer" answers than the AR semantics, and so they can also be seen as under-approximations of the latter. Investigating their complexity helps to understand whether such approximations have actually lower complexities, which is indeed the case for different languages and complexity measures considered in this paper. Second, recent work on explanation in the context of inconsistency-tolerant query answering shows that explanations are much easier to define and compute for the IAR semantics [Bienvenu *et al.*, 2016]. Third, the IAR and ICR semantics are amenable to preprocessing (the intersection can be computed offline, and then standard querying algorithms can be employed online)—indeed, this has been used to implement the IAR semantics [Lembo *et al.*, 2015], and for ICR, it has been remarked in [Bienvenu and Bourgaux, 2016].

The complexity of query answering under AR semantics when the ontology is described via one of the central DLs is well-understood. Rosati [2011] studied the data and combined complexity for a wide spectrum of DLs, while Bienvenu [2012] identified cases for simple ontologies (within the *DL-Lite* family) for which tractable data complexity results can be obtained. In [Lukasiewicz *et al.*, 2012a; 2013; 2015], the data and different types of combined complexity of the AR semantics have been studied for ontologies described via existential rules and negative constraints.

Bienvenu *et al.* [2014a] analyzed the data and the combined complexity of query answering under the AR and IAR semantics for different notions of maximal repairs over the language *DL-Lite*$_{\mathcal{R}}$. Recently, the AR semantics was extended to the *generalized repair (GR)* semantics and its computational complexity analyzed [Eiter *et al.*, 2016]. In the GR semantics, also ontological rules may be removed, and some database atoms and rules are assumed to be non-removable.

This paper continues this line of research and integrates the generalized repair semantics of [Eiter *et al.*, 2016] with the two intersection-based approximate repair semantics. We analyze the complexity of approximate inconsistency-tolerant query answering for a wide range of Datalog$^{\pm}$ languages and for several different complexity measures; in particular:

- ▷ We consider different popular inconsistency-tolerant semantics, namely, the IAR and the ICR semantics, in both their standard and their generalized repair variants.
- ▷ We consider the most popular Datalog$^{\pm}$ languages: lin-

ear, guarded, sticky, and acyclic existential rules, along with "weak" generalizations, as well as full (i.e., non-existential) restrictions, and full rules in general.

▷ We analyze the data, fixed-program combined, bounded-arity combined, and combined complexity.

2 Datalog$^{\pm}$

In this section, we briefly recall some basics on existential rules from the context of Datalog$^{\pm}$ [Calì *et al.*, 2012a].

General. We assume a set $\mathbf{C}$ of *constants*, a set $\mathbf{N}$ of *labeled nulls*, and a set $\mathbf{V}$ of *variables*. A *term* t is a constant, null, or variable. We also assume a set of *predicates*, each associated with an arity, i.e., a non-negative integer. An *atom* has the form $p(t_1, \ldots, t_n)$, where p is an n-ary predicate, and $t_1, \ldots, t_n$ are terms. Conjunctions of atoms are often identified with the sets of their atoms. An *instance* I is a (possibly infinite) set of atoms $p(\mathbf{t})$, where $\mathbf{t}$ is a tuple of constants and nulls. A *database* D is a finite instance that contains only constants. A *homomorphism* is a substitution $h\colon \mathbf{C}\cup\mathbf{N}\cup\mathbf{V} \to \mathbf{C}\cup\mathbf{N}\cup\mathbf{V}$ that is the identity on $\mathbf{C}$ and maps $\mathbf{N}$ to $\mathbf{C}\cup\mathbf{N}$. A *conjunctive query* (CQ) q has the form $\exists\mathbf{Y}\phi(\mathbf{X},\mathbf{Y})$, where $\phi(\mathbf{X},\mathbf{Y})$ is a conjunction of atoms without nulls. The *answer* to q over an instance I, denoted $q(I)$, is the set of all tuples $\mathbf{t}$ over $\mathbf{C}$ for which there is a homomorphism h such that $h(\phi(\mathbf{X},\mathbf{Y})) \subseteq I$ and $h(\mathbf{X}) = \mathbf{t}$. A *Boolean CQ* (BCQ) q is a CQ $\exists\mathbf{Y}\phi(\mathbf{Y})$, i.e., all variables are existentially quantified; q is *true* over I, denoted $I \models q$, if $q(I) \neq \emptyset$, i.e., there is a homomorphism h with $h(\phi(\mathbf{Y})) \subseteq I$.

Dependencies. A *tuple-generating dependency (TGD)* σ is a first-order formula $\forall\mathbf{X}\forall\mathbf{Y}\,\varphi(\mathbf{X},\mathbf{Y}) \rightarrow \exists\mathbf{Z}\,p(\mathbf{X},\mathbf{Z})$, where $\mathbf{X}\cup\mathbf{Y}\cup\mathbf{Z} \subseteq \mathbf{V}$, $\varphi(\mathbf{X},\mathbf{Y})$ is a conjunction of atoms, and $p(\mathbf{X},\mathbf{Z})$ is an atom, all without nulls; $\varphi(\mathbf{X},\mathbf{Y})$ is the *body* of σ, denoted $body(\sigma)$, while $p(\mathbf{X},\mathbf{Z})$ is the *head* of σ, denoted $head(\sigma)$. For clarity, we consider single-atom-head TGDs; however, our results can be extended to TGDs with a conjunction of atoms in the head. An instance I satisfies σ, written $I \models \sigma$, if the following holds: whenever there exists a homomorphism h such that $h(\varphi(\mathbf{X},\mathbf{Y})) \subseteq I$, then there exists $h' \supseteq h|_{\mathbf{X}}$, where $h|_{\mathbf{X}}$ is the restriction of h on $\mathbf{X}$, such that $h'(p(\mathbf{X},\mathbf{Z})) \in I$. A *negative constraint (NC)* ν is a first-order formula $\forall\mathbf{X}\,\varphi(\mathbf{X}) \rightarrow \bot$, where $\mathbf{X} \subseteq \mathbf{V}$, $\varphi(\mathbf{X})$ is a conjunction of atoms without nulls, called the *body* of ν, denoted $body(\nu)$, and $\bot$ denotes the truth constant *false*. An instance I satisfies ν, written $I \models \nu$, if there is no homomorphism h such that $h(\varphi(\mathbf{X})) \subseteq I$. Given a set Σ of TGDs and NCs, I satisfies Σ, written $I \models \Sigma$, if I satisfies each TGD and NC of Σ. For brevity, we omit the universal quantifiers in front of TGDs and NCs, and use the comma (instead of $\wedge$) for conjoining body atoms. Given a class of TGDs $\mathbb{C}$, we denote by $\mathbb{C}_{\bot}$ the formalism obtained by combining $\mathbb{C}$ with arbitrary NCs. Finite sets of TGDs and NCs are also called *programs*, and TGDs are also called *existential rules*.

Knowledge Bases. A *knowledge base* is a pair (D,Σ), where D is a database, and Σ is a program. For programs Σ, Σ_T and Σ_{NC} are the subsets of Σ containing the TGDs and NCs of Σ, respectively. The set of *models* of $KB = (D,\Sigma)$, denoted $mods(KB)$, is the set of instances $\{I \mid I \supseteq D \wedge I \models \Sigma\}$. We say that KB is *consistent*, if $mods(KB) \neq \emptyset$, otherwise KB is *inconsistent*. The *answer* to a CQ q relative to KB is the set of tuples $ans(q,KB) = \bigcap\{q(I) \mid I \in mods(KB)\}$. The answer to a BCQ q is *true*, denoted $KB \models q$, if $ans(q,KB) \neq \emptyset$. The decision version of the *CQ answering* problem is as follows: given a knowledge base KB, a CQ q, and a tuple of constants $\mathbf{t}$, decide whether $\mathbf{t} \in ans(q,KB)$. Since CQ answering can be reduced in LogSpace to BCQ answering, we focus on BCQs. Following Vardi (1982), the *combined complexity* of BCQ answering considers the database, the set of dependencies, and the query as part of the input. The *bounded-arity combined* (or *ba-combined*) complexity assumes that the arity of the underlying schema is bounded by an integer constant. The *fixed-program combined* (or *fp-combined*) *complexity* considers the sets of TGDs and NCs as fixed; the *data complexity* also assumes the query fixed.

The Datalog$^{\pm}$ languages that we consider to guarantee decidability are among the most frequently analyzed in the literature, namely, linear (L) [Calì *et al.*, 2012a], guarded (G) [Calì *et al.*, 2013], sticky (S) [Calì *et al.*, 2012b], and acyclic TGDs (A), along with their "weak" (proper) generalizations weakly guarded (WG) [Calì *et al.*, 2013], weakly sticky (WS) [Calì *et al.*, 2012b], and weakly acyclic TGDs (WA) [Fagin *et al.*, 2005], as well as their "full" (proper) restrictions linear full (LF), guarded full (GF), sticky full (SF), and acyclic full TGDs (AF), respectively, and full (i.e., existential-free) TGDs (F) in general. We also recall the following further inclusions: $\mathsf{L} \subset \mathsf{G}$, $\mathsf{F} \subset \mathsf{WA} \subset \mathsf{WS}$, and $\mathsf{F} \subset \mathsf{WG}$. We refer to [Eiter *et al.*, 2016] for a more detailed overview and complexity results.

Complexity Classes. We briefly recall the complexity classes that we encounter. The complexity class AC^0 is the class of all decision problems that can be solved by uniform families of Boolean circuits of polynomial size and constant depth. PSPACE (resp., P, EXP, 2EXP) is the class of all problems that can be decided in polynomial space (resp., polynomial time, exponential time, double exponential time) on a deterministic Turing machine. NP and NEXP are the classes of all problems that are decidable in polynomial and exponential time on a nondeterministic Turing machine, respectively, and co-NP and co-NEXP are their complementary classes, where 'yes' and 'no' instances are interchanged. The class Θ_2^{P} is the class of all problems that can be decided in polynomial time by a deterministic Turing machine with either a logarithmic number of calls to an NP oracle, or (equivalently) a constant number of rounds of polynomially many parallel calls to an NP oracle. P$^{\mathrm{NEXP}}$ is the class of all problems that are decidable in deterministic polynomial time using a NEXP oracle. The class Σ_2^{P} is the class of all problems that can be decided in nondeterministic polynomial time using an NP oracle, and Π_2^{P} is the complement of Σ_2^{P}. The above complexity classes and their inclusion relationships (which are all currently believed to be strict) are: AC$^0 \subseteq$ P $\subseteq$ NP, co-NP $\subseteq \Theta_2^{\mathrm{P}} \subseteq \Sigma_2^{\mathrm{P}}, \Pi_2^{\mathrm{P}} \subseteq$ PSPACE $\subseteq$ EXP $\subseteq$ NEXP, co-NEXP $\subseteq$ P$^{\mathrm{NEXP}} \subseteq$ 2EXP.

3 Approximate Inconsistency Semantics

We now recall three prominent inconsistency-tolerant semantics for ontology-based query answering, namely, the *ABox repair (AR)* semantics and its approximation by the *intersec-*

tion of repairs (IAR) and the *intersection of closed repairs (ICR)* semantics [Lembo *et al.*, 2010; Bienvenu, 2012]; all three are based on the notion of *repair*, which is a maximal consistent subset of the given database. Furthermore, we newly define generalized repair variants [Eiter *et al.*, 2016] of the two intersection-based approximate repair semantics.

Classically, errors leading to inconsistencies are assumed to be only in the database. Eiter *et al.* [2016] introduced the *generalized inconsistency semantics* allowing for errors also in the ontology, and for parts of the database and the ontology to be without errors. We analyze the aforementioned inconsistency-tolerant semantics also for this generalized framework. More specifically, for a knowledge base (D, Σ), the generalized semantics allows also (i) to minimally remove TGDs from Σ, and (ii) to partition both D and Σ into a hard and a soft part of non-removable and removable elements, respectively. The so partitioned database (resp., program) is called *flexible database* (resp., *program*).

One application of the generalized semantics is debugging mappings between distributed ontologies. In this context, every ontology for itself is error-free, whereas the mappings between the ontologies may be erroneous (e.g., as they are automatically generated). Similarly, some (e.g., manually checked) parts of the underlying databases may be without errors, while other (e.g., automatically generated) parts may contain errors. Under the generalized semantics, inconsistent distributed ontologies are repaired by removing a minimal set of database atoms and existential rules from the mappings.

Another important application is debugging ontologies that have been created in part manually (or checked manually, ensuring error-freeness) and in part enriched by automatically learned additional parts. The manually created part is modeled as the hard database and program, while the additionally learned part is the soft database and program.

Notice that NCs are non-removable (only TGDs can be removed). This is especially well-suited in the aforementioned applications, which require to repair rules, but not negative constraints (or equality-generating dependencies).

A *flexible database* is a pair (D_h, D_s) of databases, called the *hard* and *soft database*, respectively. A *flexible program* is a pair (Σ_h, Σ_s) consisting of a finite set Σ_h of TGDs and NCs and a finite set Σ_s of TGDs, called the *hard* and *soft program*, respectively. A *flexible knowledge base* is a pair $((D_h, D_s), (\Sigma_h, \Sigma_s))$, where (D_h, D_s) is a flexible database, and (Σ_h, Σ_s) is a flexible program. Note that a (standard) knowledge base (D, Σ) is a special case of a flexible one $((D_h, D_s), (\Sigma_h, \Sigma_s))$, where $D_h = \emptyset$, $D_s = D$, $\Sigma_h = \Sigma$, and $\Sigma_s = \emptyset$. Below, we provide definitions for flexible knowledge bases that generalize the ones for (standard) knowledge bases.

For knowledge bases $KB' = (D', \Sigma')$ and $KB'' = (D'', \Sigma'')$, we write $KB' \subseteq KB''$, if $D' \subseteq D''$ and $\Sigma' \subseteq \Sigma''$. A *selection* of a flexible knowledge base $((D_h, D_s), (\Sigma_h, \Sigma_s))$ is a knowledge base (D', Σ') such that $D_h \subseteq D' \subseteq (D_h \cup D_s)$ and $\Sigma_h \subseteq \Sigma' \subseteq (\Sigma_h \cup \Sigma_s)$. A *repair* of a flexible knowledge base FKB is an inclusion-maximal consistent selection of FKB. $Rep(FKB)$ denotes the set of all repairs of FKB. Notice that for (standard) knowledge bases, a repair is usually defined as a maximal consistent subset of the database. However, when a flexible knowledge base models a standard one (i.e., $D_h = \emptyset$ and $\Sigma_s = \emptyset$), the definition above coincides with the classical one and the difference is only notational.

Example 1. Consider the flexible database (D_h, D_s) given by

$$D_h = \{Postdoc(p), Researcher(p), leaderOf(p', g')\},$$
$$D_s = \{Prof(p), leaderOf(p, g)\},$$

asserting that p is a postdoc, a researcher, a professor, and the leader of the research group g, and that p' is the leader of g'.

Consider also the flexible program (Σ_h, Σ_s) defined as

$$\begin{array}{lllll} \Sigma_h = & \{Prof(X) & \rightarrow & Researcher(X), \\ & Postdoc(X) & \rightarrow & Researcher(X), \\ & Prof(X), Postdoc(X) & \rightarrow & \bot, \\ & leaderOf(X, Y) & \rightarrow & Group(Y)\}, \\ \Sigma_s = & \{leaderOf(X, Y) & \rightarrow & Prof(X)\}, \end{array}$$

expressing that professors and postdocs are researchers, professors and postdocs form disjoint sets, and $leaderOf$ has $Prof$ as domain and $Group$ as range. It is easy to see that $mods(D, \Sigma) = \emptyset$, since p violates the disjointness constraint.

The flexible knowledge base $((D_h, D_s), (\Sigma_h, \Sigma_s))$ has two repairs (D', Σ') and (D'', Σ''):

$$\begin{array}{lllllll} D' & = & D_h \cup \{leaderOf(p, g)\}, & & \Sigma' & = & \Sigma_h, \\ D'' & = & D_h, & & \Sigma'' & = & \Sigma_h \cup \Sigma_s. \end{array}$$

In both, the atom $Prof(p)$ is removed; in the first one, also the rule $leaderOf(X, Y) \rightarrow Prof(X)$ is removed, while in the second one, the atom $leaderOf(p, g)$ is removed. ∎

We now define the inconsistency-tolerant semantics considered. For a knowledge base $KB = (D, \Sigma)$, the *closure* $Cn(KB)$ of KB is the set of all ground atoms, built from constants in D and Σ, entailed by D and the TGDs of Σ. Let FKB be a flexible knowledge base, and let q be a BCQ.

- FKB entails q under the *generalized repair (GR) semantics*, if, for all $KB' \in Rep(FKB)$, $KB' \models q$.
- FKB entails q under the *generalized intersection of repairs (GIAR) semantics*, if $(D^*, \Sigma^*) \models q$, where $D^* = \bigcap\{D' \mid (D', \Sigma') \in Rep(FKB)\}$ and $\Sigma^* = \bigcap\{\Sigma' \mid (D', \Sigma') \in Rep(FKB)\}$.
- FKB entails q under the *generalized intersection of closed repairs (GICR) semantics*, if $(D_I, \Sigma^*) \models q$, where $D_I = \bigcap\{Cn(KB') \mid KB' \in Rep(FKB)\}$ and $\Sigma^* = \bigcap\{\Sigma' \mid (D', \Sigma') \in Rep(FKB)\}$.

In the definition above, observe that if FKB is a standard knowledge base, then $\Sigma^* = \Sigma$, and thus the definition above generalizes the AR, IAR, and ICR semantics for standard knowledge bases to the case of flexible knowledge bases.

We talk of BCQ answering under the GR, GIAR, and GICR semantics when flexible knowledge bases can be arbitrary, and we talk of BCQ answering under the AR, IAR, and ICR semantics when flexible knowledge bases are restricted to model standard knowledge bases (i.e., $D_h = \emptyset$ and $\Sigma_s = \emptyset$).

4 Complexity Results

We give a precise picture of the complexity of BCQ answering from existential rules under the IAR, ICR, GIAR, and GICR semantics, which is summarized in Fig. 1; it ranges from membership in AC^0 to 2EXP-completeness.

	Data	*fp*-comb.	*ba*-comb.	Comb.
$\mathsf{L}_\perp$, $\mathsf{LF}_\perp$, $\mathsf{AF}_\perp$	in AC^{0+}	NP	Π_2^P	PSPACE
$\mathsf{S}_\perp$, $\mathsf{SF}_\perp$	in AC^{0+}	NP	Π_2^P	EXP
$\mathsf{F}_\perp$, $\mathsf{GF}_\perp$	co-NP	Θ_2^P	Π_2^P	EXP
$\mathsf{G}_\perp$	co-NP^+	Θ_2^P	EXP	2EXP
$\mathsf{A}_\perp$	in $\text{AC}^{0\star}$	NP	P^NEXP	P^NEXP
$\mathsf{WS}_\perp$, $\mathsf{WA}_\perp$	co-NP^+	Θ_2^P	2EXP	2EXP
$\mathsf{WG}_\perp$	EXP	EXP	EXP	2EXP

	Data	*fp*-comb.	*ba*-comb.	Comb.
$\mathsf{L}_\perp$, $\mathsf{LF}_\perp$, $\mathsf{AF}_\perp$	co-NP^+	Θ_2^P	Π_2^P	PSPACE
$\mathsf{S}_\perp$, $\mathsf{SF}_\perp$	co-NP^+	Θ_2^P	Π_2^P	EXP
$\mathsf{F}_\perp$, $\mathsf{GF}_\perp$	co-NP	Θ_2^P	Π_2^P	EXP
$\mathsf{G}_\perp$	co-NP^+	Θ_2^P	EXP	2EXP
$\mathsf{A}_\perp$	co-NP	Θ_2^P	P^NEXP	P^NEXP
$\mathsf{WS}_\perp$, $\mathsf{WA}_\perp$	co-NP^+	Θ_2^P	2EXP	2EXP
$\mathsf{WG}_\perp$	EXP	EXP	EXP	2EXP

Figure 1: Complexity of IAR and GIAR (left) and of ICR and GICR (right) BCQ answering; all entries without "in" are completeness results. $^+$[Lukasiewicz *et al.*, 2013] for $\mathsf{L}_\perp$, $\mathsf{S}_\perp$, $\mathsf{G}_\perp$, $\mathsf{WS}_\perp$, and $\mathsf{WA}_\perp$. *[Lukasiewicz *et al.*, 2012b].

In detail, ICR- and GICR-BCQ answering (Fig. 1, right side) is complete for co-NP (resp., Θ_2^P) in the data (resp., *fp*-combined) complexity for all languages of existential rules, but for $\mathsf{WG}_\perp$, where it is EXP-complete. The combined complexity of ICR- and GICR-BCQ answering is among PSPACE (for $\mathsf{L}_\perp$, $\mathsf{LF}_\perp$, and $\mathsf{AF}_\perp$), EXP (for $\mathsf{S}_\perp$, $\mathsf{SF}_\perp$, $\mathsf{F}_\perp$, and $\mathsf{GF}_\perp$), P^NEXP (for $\mathsf{A}_\perp$), and 2EXP (for $\mathsf{G}_\perp$, $\mathsf{WS}_\perp$, $\mathsf{WA}_\perp$, and $\mathsf{WG}_\perp$), while the *ba*-combined complexity is among Π_2^P (for $\mathsf{L}_\perp$, $\mathsf{LF}_\perp$, $\mathsf{AF}_\perp$, $\mathsf{S}_\perp$, $\mathsf{SF}_\perp$, $\mathsf{F}_\perp$, and $\mathsf{GF}_\perp$), EXP (for $\mathsf{G}_\perp$ and $\mathsf{WG}_\perp$), P^NEXP (for $\mathsf{A}_\perp$), and 2EXP (for $\mathsf{WS}_\perp$ and $\mathsf{WA}_\perp$). So, the complexity of ICR- and GICR-BCQ answering nearly coincides with the complexity of AR- and GR-BCQ answering [Eiter *et al.*, 2016], except for the Θ_2^P (rather than Π_2^P) entries. The complexity of IAR- and GIAR-BCQ answering (Fig. 1, left side) slightly drops to AC^0 and NP in the data and *fp*-combined complexity, resp., for the first-order rewritable languages (i.e., for $\mathsf{L}_\perp$, $\mathsf{LF}_\perp$, $\mathsf{AF}_\perp$, $\mathsf{S}_\perp$, $\mathsf{SF}_\perp$, and $\mathsf{A}_\perp$).

The EXP- and 2EXP-hardness results for $\mathsf{WG}_\perp$ are immediate by the EXP- and 2EXP-hardness of standard BCQ answering in these cases. In the rest of this section, we give proof sketches and ideas for some of the other results.

4.1 Membership Results

IAR semantics. For a knowledge base (D, Σ), where $\Sigma = \Sigma_T \cup \Sigma_{NC}$ is over a Datalog$^\pm$ language whose BCQ answering is in $\mathbf{C}$, we can decide in $\text{NP}^\mathbf{C}$ that (D, Σ) does *not* entail a BCQ q under IAR semantics by guessing and checking that there exist: (1) a database $D^\star \subseteq D$ with $(D^\star, \Sigma_T) \not\models q$, and (2) repairs D'_α with $\alpha \notin D'_\alpha$, one for each $\alpha \in D \setminus D^\star$ (witnessing that the intersection of the repairs is a subset of $D^\star$).

This proves all upper bounds equal to and above co-NP in Fig. 1, left side, excluding the Θ_2^P memberships.

Theorem 2. *If BCQ answering from databases under programs over some Datalog$^\pm$ language L is in $\mathbf{C}$ in the data (resp., fp-combined, ba-combined, and combined) complexity, then IAR-BCQ answering from databases under programs over L is in co-$\text{NP}^\mathbf{C}$ in the data (resp., fp-combined, ba-combined, and combined) complexity.*

Consider the Datalog$^\pm$ languages whose BCQ answering in the data complexity is in AC^0, i.e., $\mathsf{L}_\perp$, $\mathsf{S}_\perp$, $\mathsf{A}_\perp$, $\mathsf{LF}_\perp$, $\mathsf{AF}_\perp$, and $\mathsf{SF}_\perp$. To evaluate a BCQ under the IAR semantics, in the *fp*-combined complexity, the NCs are fixed and correspond to a fixed UCQ over the database. We can thus compute all ground instances of conjunctions, and calculate all minimal ones in polynomial time. We then remove all their atoms from the database, producing the intersection of all repairs, which we use to evaluate the query, which is in NP for $\mathsf{L}_\perp$, $\mathsf{S}_\perp$, and $\mathsf{A}_\perp$ (and so also for $\mathsf{LF}_\perp$, $\mathsf{AF}_\perp$, and $\mathsf{SF}_\perp$); this is thus also the overall upper bound in the *fp*-combined complexity.

Theorem 3. *IAR-BCQ answering for $\mathsf{L}_\perp$, $\mathsf{S}_\perp$, and $\mathsf{A}_\perp$ (and $\mathsf{LF}_\perp$, $\mathsf{AF}_\perp$, and $\mathsf{SF}_\perp$) is in NP in the fp-combined complexity.*

Consider next the Datalog$^\pm$ languages whose BCQ answering in the data complexity is in P. In the *fp*-combined complexity, checking that a selection of a knowledge base in $\mathsf{WS}_\perp$ or $\mathsf{G}_\perp$ (and thus also in $\mathsf{WA}_\perp$, $\mathsf{F}_\perp$, or $\mathsf{GF}_\perp$) is a repair is feasible in polynomial time, because the TGDs and NCs are fixed, and answering BCQs in the data complexity and atomic queries in the *fp*-combined complexity for these languages is in P. A P machine can compute the intersection of the repairs by asking in parallel to polynomially many NP oracles whether there exists a repair that excludes a specific ground atom. Once the intersection is computed, an additional NP oracle call allows to decide whether the intersection of repairs entails the query—we recall that BCQ answering in the *fp*-combined complexity for the $\mathsf{WS}_\perp$ and $\mathsf{G}_\perp$ (and thus also for the $\mathsf{WA}_\perp$, $\mathsf{F}_\perp$, and $\mathsf{GF}_\perp$) languages is in NP. Overall, BCQ query answering under the IAR semantics is in Θ_2^P, which proves all Θ_2^P upper bounds in Fig. 1, left side.

Theorem 4. *IAR-BCQ answering for $\mathsf{WS}_\perp$ and $\mathsf{G}_\perp$ (and $\mathsf{WA}_\perp$, $\mathsf{F}_\perp$, and $\mathsf{GF}_\perp$) is in Θ_2^P in the fp-combined complexity.*

ICR semantics. The following theorem shows that ICR-BCQ answering for a Datalog$^\pm$ language L is in co-$\text{NP}^\mathbf{C}$ in the data, *fp*-combined, and *ba*-combined complexity, where $\mathbf{C}$ is an upper bound for BCQ answering for L. It can be shown similarly to Theorem 2: we guess and verify some $D^\star \subseteq Cn(D, \Sigma_T)$ with $(D^\star, \Sigma_T) \not\models q$, and, for each (of the polynomially many) $\alpha \in Cn(D, \Sigma_T) \setminus D^\star$, some repair D'_α with $\alpha \notin Cn(D'_\alpha, \Sigma_T)$. This proves all corresponding upper bounds in Fig. 1, right side, including the $\text{P}^\text{NEXP} = \text{co-NP}^\text{NEXP}$membership for $\mathsf{A}_\perp$, excluding memberships in Θ_2^P.

Theorem 5. *If BCQ answering from databases under programs over some Datalog$^\pm$ language L is in $\mathbf{C}$ in the data (resp., fp- and ba-combined) complexity, then ICR-BCQ answering from databases under programs over L is in co-$\text{NP}^\mathbf{C}$ in the data (resp., fp- and ba-combined) complexity.*

ICR-BCQ answering for all the considered Datalog$^\pm$ languages but $\mathsf{WG}_\perp$ is in Θ_2^P in the *fp*-combined complexity. This is obtained by adapting the proof of Theorem 4: a P machine computes the intersection of closed repairs by asking in parallel to its NP oracle whether a ground atom is not in the closure of a repair. Being the program fixed, there are only

polynomially many different ground atoms to consider. This proves all Θ_2^{p} upper bounds in Fig. 1, right side.

Theorem 6. *ICR-BCQ answering for all the considered Datalog$^{\pm}$ languages but* $\mathsf{WG}_\bot$ *is in* Θ_2^{p} *in the fp-combined complexity.*

As for the combined complexity of ICR-BCQ answering, let (D, Σ) be a knowledge base with Σ over a Datalog$^{\pm}$ language whose BCQ reasoning is in a deterministic complexity class $\mathbf{C}$. We can decide in PSPACE$\cdot\mathbf{C}$ that (D, Σ) does *not* entail a BCQ q by checking, for all ground instances q' of q, that there is a repair $D^\star \subseteq D$ such that $(D^\star, \Sigma_T) \not\models q'$. This proves all corresponding upper bounds in Fig. 1, right side, including the $\text{P}^{\text{NEXP}} = \text{PSPACE} \cdot \text{P}^{\text{NEXP}}$ membership for $\mathsf{A}_\bot$,

Theorem 7. *If BCQ answering from databases under programs over some Datalog$^{\pm}$ language L is in the deterministic complexity class* $\mathbf{C}$ *in the combined complexity, then ICR-BCQ answering from databases under programs over L is in* PSPACE $\cdot$ $\mathbf{C}$ *in the combined complexity.*

Generalized repair semantics. All upper bounds for IAR (resp., ICR) BCQ answering from databases D under Datalog$^{\pm}$ programs Σ carry over to GIAR (resp., GICR) BCQ answering from databases (D_h, D_s) under Datalog$^{\pm}$ programs $(\Sigma_h, \emptyset)$. Based on this, also all membership results for IAR (resp., ICR) BCQ answering under existential rules carry over to GIAR (resp., GICR) BCQ answering, as long as the existential rules are closed under adding 0-ary body atoms, proving all corresponding upper bounds in Fig. 1 but for the linear cases (for which it is not hard to derive the upper bounds by genuine proofs: rules in the repair can be polynomially guessed, like data).

Theorem 8. *Let L be a Datalog$^{\pm}$ language that is closed under adding 0-ary atoms to rule bodies. If IAR (resp., ICR) BCQ answering from databases and programs over L is in* $\mathbf{C}$ *in the data, combined, and ba- and fp-combined complexity, then GIAR (resp., GICR) BCQ answering from flexible databases and programs over L is also in* $\mathbf{C}$ *in the data, combined, and ba- and fp-combined complexity, respectively.*

4.2 Hardness Results

As BCQ answering under the IAR and ICR semantics for Datalog$^{\pm}$ languages L coincides with BCQ answering for L when there are no inconsistencies, we immediately obtain hardness for all NP, PSPACE, EXP, and 2EXP entries in Fig. 1.

IAR semantics. co-NP-hardness of BCQ answering under IAR semantics in the data complexity is shown by a reduction from unsatisfiability of 3CNF formulas. It produces a knowledge base with a fixed $\mathsf{GF}_\bot$ program and fixed query. This proves all open co-NP-hardness results in Fig. 1, left side.

Theorem 9. *IAR-BCQ answering for* $\mathsf{GF}_\bot$ *(and* $\mathsf{F}_\bot$*) is* co-NP*-hard in the data complexity.*

The following result shows that IAR-BCQ answering for $\mathsf{A}_\bot$ is P^{NEXP}-hard in the *ba*-combined complexity, proving all P^{NEXP}-hardness results in Fig. 1, left side. Intuitively, the reduction for the P^{NEXP}-hardness proof in [Eiter *et al.*, 2016] for AR-BCQ answering for $\mathsf{A}_\bot$ in the *ba*-combined complexity is turned into a P^{NEXP}-hardness proof for IAR-BCQ answering in this case. There, one encodes initial tiling assignments $v_1(X_i), \ldots, v_n(X_n)$ and has a ground atomic query q, which we now also include in the database along with a fresh ground atom nq and the NC $v_1(X_i) \wedge \ldots \wedge v_n(X_n) \wedge q \wedge nq \rightarrow \bot$. This intuitively "forces" the atom q into the database.

Theorem 10. *IAR-BCQ answering for* $\mathsf{A}_\bot$ *is* P^{NEXP}*-hard in the ba-combined complexity.*

The next result shows that BCQ answering under the IAR semantics for $\mathsf{GF}_\bot$ (and thus also for $\mathsf{F}_\bot$, $\mathsf{G}_\bot$, $\mathsf{WA}_\bot$, and $\mathsf{WS}_\bot$) in the *fp*-combined complexity is Θ_2^{p}-hard. A sketch of the technically quite involved proof is given below.

Theorem 11. *IAR-BCQ answering for* $\mathsf{GF}_\bot$ *(and* $\mathsf{F}_\bot$*,* $\mathsf{G}_\bot$*,* $\mathsf{WA}_\bot$*, and* $\mathsf{WS}_\bot$*) is* Θ_2^{p}-hard *in the fp-combined complexity.*

Proof (sketch). We show a reduction from the Θ_2^{p}-complete problem COMP-SAT: for two sets A and B of 3CNF formulas, decide whether A contains more satisfiable formulas than B [Lukasiewicz and Malizia, 2016; 2017]. COMP-SAT is Θ_2^{p}-hard even if its instances (A, B) are restricted to be such that $|A| = |B|$, all formulas in A and B are defined over the same variables and have the same number of clauses, and $A = \{\phi_1, \ldots, \phi_t\}$ and $B = \{\psi_1, \ldots, \psi_t\}$ are such that ϕ_{u+1} (resp., ψ_{u+1}) being satisfiable implies ϕ_u (resp., ψ_u) being satisfiable, for any u (intuitively, all satisfiable formulas have the lowest indices in sets A and B). From this, it follows that (A, B) is a 'yes'-instance of COMP-SAT iff there is an index u such that $\phi_u \in A$ is satisfiable, and $\psi_u \in B$ is not.

From an instance (A, B) of COMP-SAT, we build the knowledge base $KB_{CS1}((A, B)) = (D_{CS1}, \Sigma_{CS1})$ as follows. Let us assume that all formulas in sets A and B are defined over variables $X = \{x_1, \ldots, x_n\}$ and clauses $C = \{c_1, \ldots, c_m\}$, with $\ell_{j,k}$ denoting the k^{th} literal of the j^{th} clause, and $v_{j,k}$ denoting the variable of literal $\ell_{j,k}$.

Let us see the encoding of formulas $\psi_u(X) \in B$. For each variable $x_i \in X$, there are facts $Val(u, x_i, t)$ and $Val(u, x_i, f)$ in D_{CS1}, where u, x_i, t, and f are constants referring to the formula number u, variable x_i, and the Boolean values *true* and *false*, respectively. Constant u in the atoms above and below is required so that atoms referring to different formulas $\psi_u \in B$ do not interfere. For each clause c_j:

- there is a fact $SuccCl(u, j-1, j)$ in D_{CS1}, where $j-1$ and j are numeric constants, and intuitively stating that the j^{th} clause is the successor of the $(j-1)^{\text{th}}$ clause; and
- there is a fact encoding c_j; e.g., for a clause $(x_p \vee x_q \vee \neg x_r)$, there is the fact $Cl(u, j, x_p, p, x_q, p, x_r, n)$ in D_{CS1}, where p and n are constants telling whether a literal appears positively or negatively, respectively.

Three extra facts are in D_{CS1}: $SatChain(u, 0)$, $MaxCl(u, m)$, and $Unsat(u)$, where m is a numeric constant for the number of the formula's clauses. Rules of KB_{CS1} are:

$$Val(U, X, t),\ Val(U, X, f) \rightarrow \bot$$
$$Sat(U),\ Unsat(U) \rightarrow \bot$$
$$Cl(U, J, X, p, \_, \_, \_, \_),\ Val(U, X, t) \rightarrow SatCl(U, J)$$
$$Cl(U, J, X, n, \_, \_, \_, \_),\ Val(U, X, f) \rightarrow SatCl(U, J)$$
$$Cl(U, J, \_, \_, Y, p, \_, \_),\ Val(U, Y, t) \rightarrow SatCl(U, J)$$
$$Cl(U, J, \_, \_, Y, n, \_, \_),\ Val(U, Y, f) \rightarrow SatCl(U, J)$$

$$Cl(U, J, \_, \_, \_, \_, Z, p), Val(U, Z, t) \rightarrow SatCl(U, J)$$
$$Cl(U, J, \_, \_, \_, \_, Z, n), Val(U, Z, f) \rightarrow SatCl(U, J)$$
$$SatChain(U, I), SuccCl(U, I, J), SatCl(U, J) \rightarrow SatChain(U, J)$$
$$MaxCl(U, M), SatChain(U, M) \rightarrow Sat(U).$$

Let us now see how to encode (the satisfiability) of formulas $\phi_u(X) \in A$ via facts in the database and a specific query. For each clause c_j of $\phi_u(X)$, in the database D_{CS1}, there are facts encoding c_j along with all the possible assignments satisfying c_j. For example, for a clause $(x_p \vee x_q \vee \neg x_r)$, there are 7 facts, each of them for a possible way of satisfying c_j:

$$Cl_j(u, x_p, f, x_q, f, \neg x_r, t) \quad Cl_j(u, x_p, t, x_q, f, \neg x_r, t)$$
$$Cl_j(u, x_p, f, x_q, t, \neg x_r, f) \quad Cl_j(u, x_p, t, x_q, t, \neg x_r, f)$$
$$Cl_j(u, x_p, f, x_q, t, \neg x_r, t) \quad Cl_j(u, x_p, t, x_q, t, \neg x_r, t)$$
$$Cl_j(u, x_p, t, x_q, f, \neg x_r, f),$$

where u, x_p, x_q, $\neg x_r$, t, and f are constants referring to formula number u, literals x_p, x_q, and $\neg x_r$, and the Boolean values $true$ and $false$, respectively. Predicates encoding clauses of A's formulas are distinct from those of B's formulas.

Furthermore, in D_{CS1}, there are specific facts to enforce consistent assignments when recognizing satisfiable formulas in A. In particular, for each variable x_i, there are 8 facts:

$$Cons(x_i, f, x_i, f) \quad Cons(x_i, t, \neg x_i, f) \quad Cons(\neg x_i, f, \neg x_i, f)$$
$$Cons(x_i, t, x_i, t) \quad Cons(\neg x_i, f, x_i, t) \quad Cons(\neg x_i, t, \neg x_i, t)$$
$$Cons(x_i, f, \neg x_i, t) \quad Cons(\neg x_i, t, x_i, f).$$

We define the query by showing its parts. A part enforces that assignments selected to satisfy formulas in A are consistent:

$$Consistency \equiv \bigwedge_{\substack{\forall (\ell_{j,k}, \ell_{j',k'}) \\ \text{s.t. } v_{j,k} = v_{j',k'}}} Cons(L_{j,k}, T_{j,k}, L_{j',k'}, T_{j',k'}).$$

Another query's part identifies satisfiable formulas in A:

$$Clauses \equiv \bigwedge_{j=1}^{m} Cl_j(U, L_{j,1}, T_{j,1}, L_{j,2}, T_{j,2}, L_{j,3}, T_{j,3}).$$

The query is $q_{CS1} = (\exists U, L_{1,1}, T_{1,1}, \ldots, L_{m,3}, T_{m,3})$ $(Clauses \wedge Consistency \wedge Unsat(U))$. The TGDs do not depend on (A, B). $KB_{CS1}(\phi)$ is guarded and full.

It is possible to show that there is an index u, for which $\phi_u \in A$ is satisfiable and $\psi_u \in B$ is unsatisfiable, iff KB_{CS1} entails q_{CS1} under the AR, ICR, and IAR semantics. □

ICR semantics. ICR-BCQ answering in the data complexity is co-NP by a reduction unsatisfiability of 3CNF formulas. The reduction produces a knowledge base with a fixed $\mathsf{LF}_\bot$, $\mathsf{AF}_\bot$, and $\mathsf{SF}_\bot$ program and a fixed query. This proves all open co-NP-hardness results in Fig. 1, right side.

Theorem 12. *ICR-BCQ answering for $\mathsf{LF}_\bot$, $\mathsf{AF}_\bot$, and $\mathsf{SF}_\bot$ (and $\mathsf{GF}_\bot$, $\mathsf{F}_\bot$, and $\mathsf{A}_\bot$) is* co-NP*-hard in the data complexity.*

BCQ answering under the ICR semantics is Θ_2^{p}-hard in the *fp*-combined complexity for all remaining entries in Fig. 1, right side, except for $\mathsf{WG}_\bot$. The proof adapts the reduction used in Theorem 11 with the reduction of Theorem 12.

Theorem 13. *ICR-BCQ answering in the fp-combined complexity is* Θ_2^{p}-hard *for all the considered Datalog$^\pm$ languages.*

BCQ answering under the ICR semantics for $\mathsf{AF}_\bot$ (and so also for $\mathsf{F}_\bot$) is Π_2^{p}-hard in the ba-combined complexity.

Theorem 14. *ICR-BCQ answering is* Π_2^{p}-hard *in the ba-combined complexity for $\mathsf{AF}_\bot$ (and $\mathsf{F}_\bot$).*

Proof (sketch). The hardness result is proven via a reduction from the Π_2^{p}-complete problem $NQBF_{2,\forall}$ [Greco *et al.*, 2011; Schaefer, 2001]: decide the validity of a formula $\Phi = (\forall X)(\exists Y)\phi(X, Y)$, where $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_r\}$, and $\phi(X, Y) = c_{i(1)} \wedge c_{\bar{i}(1)} \wedge \cdots \wedge c_{i(n)} \wedge c_{\bar{i}(n)} \wedge c_1 \wedge \cdots \wedge c_m$ is a 3CNF formula, where each $x_k \in X$ occurs only in the two clauses $c_{i(k)} = (x_k \vee \neg y_k)$ and $c_{\bar{i}(k)} = (\neg x_k \vee y_k)$—intuitively, each variable x_k enforces the truth value of the variable y_k.

The key ideas of the reduction are as follows. Repairs encode the assignments over X. The satisfiability of $\phi(X, Y)$ is checked through a TGD in which, via tailored facts in the knowledge base, we impose that the assignments over Y are consistent and they match the assignments over X. The predicate in the TGD's head can be derived iff $\phi(X, Y)$ can be satisfied given the assignment over X encoded in the repair. The TGD's head can be derived in all repairs iff Φ is valid. □

The following shows that ICR-BCQ answering for $\mathsf{A}_\bot$ is $\mathrm{P}^{\mathrm{NEXP}}$-hard in the ba-combined complexity, proving all $\mathrm{P}^{\mathrm{NEXP}}$-hardness results in Fig. 1, right side. This follows from the fact that ICR-BCQ answering coincides with AR-BCQ answering for ground BCQs, and that the $\mathrm{P}^{\mathrm{NEXP}}$-hardness proof in [Eiter *et al.*, 2016] for AR-BCQ answering for $\mathsf{A}_\bot$ in the ba-combined complexity only uses a ground atomic query.

Theorem 15. *ICR-BCQ answering for $\mathsf{A}_\bot$ is $\mathrm{P}^{\mathrm{NEXP}}$-hard in the ba-combined complexity.*

The following shows all Π_2^{p}-hardness results in Fig. 1, right side. It is proved by adapting the proof of Theorem 14, where the TGD is encoded as two negative constraints. The Π_2^{p}-hardness results in Fig. 1, left side, are proved similarly.

Theorem 16. *ICR-BCQ answering for $\mathsf{L}_\bot$, $\mathsf{LF}_\bot$, $\mathsf{AF}_\bot$, $\mathsf{S}_\bot$, $\mathsf{SF}_\bot$, $\mathsf{F}_\bot$, and $\mathsf{GF}_\bot$ is Π_2^{p}-hard in the ba-combined complexity.*

Generalized repair semantics. The next result shows that all hardness results for IAR- and ICR-BCQ answering under the different classes of existential rules carry over to GIAR- and GICR-BCQ answering, proving all lower bounds in Fig. 1.

Theorem 17. *If IAR (resp., ICR) BCQ answering from databases and programs over a Datalog$^\pm$ language L is* **C***-hard in the data, combined, and ba- and fp-combined complexity, then GIAR (resp., GICR) BCQ answering from flexible databases and programs over L is also* **C***-hard in the data, combined, and ba- and fp-combined complexity, respectively.*

5 Summary and Outlook

We have given a precise picture of the complexity of BCQ answering under different approximate inconsistency-tolerant semantics (namely, the intersection of repairs (IAR) and the intersection of closed repairs (ICR) semantics) for the most popular Datalog$^\pm$ languages and complexity measures. In addition to the standard setting, where only database atoms can be removed, we have also considered the more general

setting, where also rules may be removed (called generalized IAR (GIAR) and generalized ICR (GICR) semantics).

Future research lines include considering other classes of existential rules and to define other semantics for inconsistency-tolerant ontological query answering. In particular, it would be interesting to explore whether there are data-tractable and/or even first-order rewritable other such semantics. Furthermore, a more fine-grained way to analyze the complexity of query answering would be a non-uniform approach, looking at the complexity of a single ontology or a single ontology-mediated query (see, e.g., [Bienvenu *et al.*, 2014b; Koutris and Suciu, 2014; Hernich *et al.*, 2017]).

Acknowledgments

This work was supported by The Alan Turing Institute under the UK EPSRC grant EP/N510129/1, and by the EPSRC grants EP/R013667/1, EP/L012138/1, and EP/M025268/1.

References

[Arenas *et al.*, 1999] M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *Proc. PODS*, pp. 68–79, 1999.

[Bienvenu and Bourgaux, 2016] M. Bienvenu and C. Bourgaux. Inconsistency-tolerant querying of description logic knowledge bases. In *Reasoning Web*, pp. 156–202, 2016.

[Bienvenu and Rosati, 2013] M. Bienvenu and R. Rosati. Tractable approximations of consistent query answering for robust ontology-based data access. In *Proc. IJCAI*, pp. 775–781, 2013.

[Bienvenu *et al.*, 2014a] M. Bienvenu, C. Bourgaux, and F. Goasdoué. Querying inconsistent description logic knowledge bases under preferred repair semantics. In *Proc. AAAI*, pp. 996–1002, 2014.

[Bienvenu *et al.*, 2014b] M. Bienvenu, B. ten Cate, C. Lutz, and F. Wolter. Ontology-based data access: A study through disjunctive Datalog, CSP, and MMSNP. *ACM Trans. Database Syst.*, 39(4):33:1–33:44, 2014.

[Bienvenu *et al.*, 2016] M. Bienvenu, C. Bourgaux, and F. Goasdoué. Explaining inconsistency-tolerant query answering over description logic knowledge bases. In *Proc. AAAI*, pp. 900–906, 2016.

[Bienvenu, 2012] M. Bienvenu. On the complexity of consistent query answering in the presence of simple ontologies. In *Proc. AAAI*, pp. 705–711, 2012.

[Calì *et al.*, 2012a] A. Calì, G. Gottlob, and T. Lukasiewicz. A general Datalog-based framework for tractable query answering over ontologies. *J. Web Sem.*, 14:57–83, 2012.

[Calì *et al.*, 2012b] A. Calì, G. Gottlob, and A. Pieris. Towards more expressive ontology languages: The query answering problem. *Artif. Intell.*, 193:87–128, 2012.

[Calì *et al.*, 2013] A. Calì, G. Gottlob, and M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res.*, 48:115–174, 2013.

[Eiter *et al.*, 2016] T. Eiter, T. Lukasiewicz, and L. Predoiu. Generalized consistent query answering under existential rules. In *Proc. KR*, pp. 359–368, 2016.

[Fagin *et al.*, 2005] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.

[Greco *et al.*, 2011] G. Greco, E. Malizia, L. Palopoli, and F. Scarcello. On the complexity of core, kernel, and bargaining set. *Artif. Intell.*, 175(12/13):1877–1910, 2011.

[Hernich *et al.*, 2017] A. Hernich, C. Lutz, F. Papacchini, and F. Wolter. Dichotomies in ontology-mediated querying with the guarded fragment. In *Proc. PODS*, pp. 185–199, 2017.

[Koutris and Suciu, 2014] P. Koutris and D. Suciu. A dichotomy on the complexity of consistent query answering for atoms with simple keys. In *Proc. ICDT*, pp. 165–176, 2014.

[Lembo *et al.*, 2010] D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, and D. F. Savo. Inconsistency-tolerant semantics for description logics. In *Proc. RR*, pp. 103–117, 2010.

[Lembo *et al.*, 2015] D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, and D. F. Savo. Inconsistency-tolerant query answering in ontology-based data access. *J. Web Sem.*, 33:3–29, 2015.

[Lukasiewicz and Malizia, 2016] T. Lukasiewicz and E. Malizia. On the complexity of mCP-nets. In *Proc. AAAI*, pp. 558–564, 2016.

[Lukasiewicz and Malizia, 2017] T. Lukasiewicz and E. Malizia. A novel characterization of the complexity class Θ_k^P based on counting and comparison. *Theor. Comput. Sci.*, 694:21–33, 2017.

[Lukasiewicz *et al.*, 2012a] T. Lukasiewicz, M. V. Martinez, and G. I. Simari. Inconsistency handling in Datalog+/– ontologies. In *Proc. ECAI*, pp. 558–563, 2012.

[Lukasiewicz *et al.*, 2012b] T. Lukasiewicz, M. V. Martinez, and G. I. Simari. Inconsistency-tolerant query rewriting for linear Datalog+/–. In *Datalog 2.0*, pp. 123–134, 2012.

[Lukasiewicz *et al.*, 2013] T. Lukasiewicz, M. V. Martinez, and G. I. Simari. Complexity of inconsistency-tolerant query answering in Datalog+/–. In *Proc. OTM*, pp. 488–500, 2013.

[Lukasiewicz *et al.*, 2015] T. Lukasiewicz, M. V. Martinez, A. Pieris, and G. I. Simari. From classical to consistent query answering under existential rules. In *Proc. AAAI*, pp. 1546–1552, 2015.

[Rosati, 2011] R. Rosati. On the complexity of dealing with inconsistency in description logic ontologies. In *Proc. IJCAI*, pp. 1057–1062, 2011.

[Schaefer, 2001] M. Schaefer. Graph Ramsey theory and the polynomial hierarchy. *J. Comput. Syst. Sci.*, 62(2):290–322, 2001.

[Vardi, 1982] M. Y. Vardi. The complexity of relational query languages (extended abstract). In *Proc. STOC*, pp. 137–146, 1982.